Introduction à la sécurité informatique - 0
Crypto 101

F. Prost
Frederic.Prost@univ-grenoble-alpes.fr

UGA

18 Janvier 2023

Les solutions cryptographiques courantes en 2023

- La cryptographie est passée de l'artisanat à l'industrie dans les années 90.

  avant l'intérêt était limité (peu d'ordinateurs, pas de réseau internet largement disponible, difficultés techniques des écoutes etc.) et toutes sortes de limitations légales (un algo de cryptage était vu comme une arme de guerre comme un porte-avion ou un char d'assaut).
- Grande action de standardisation de la NSA (son but premier en fait).
- Les algos sont publics. Seul endroit où il faut garder le secret est limité à la clef.
- Plusieurs familles d'algorithmes :
  - Symétriques / asymétriques.
  - Par bloc / stream.
- Problèmes :
  - symétrique : gestion des clefs.
  - asymétrique : lenteur (facteur 1000 à 1000)

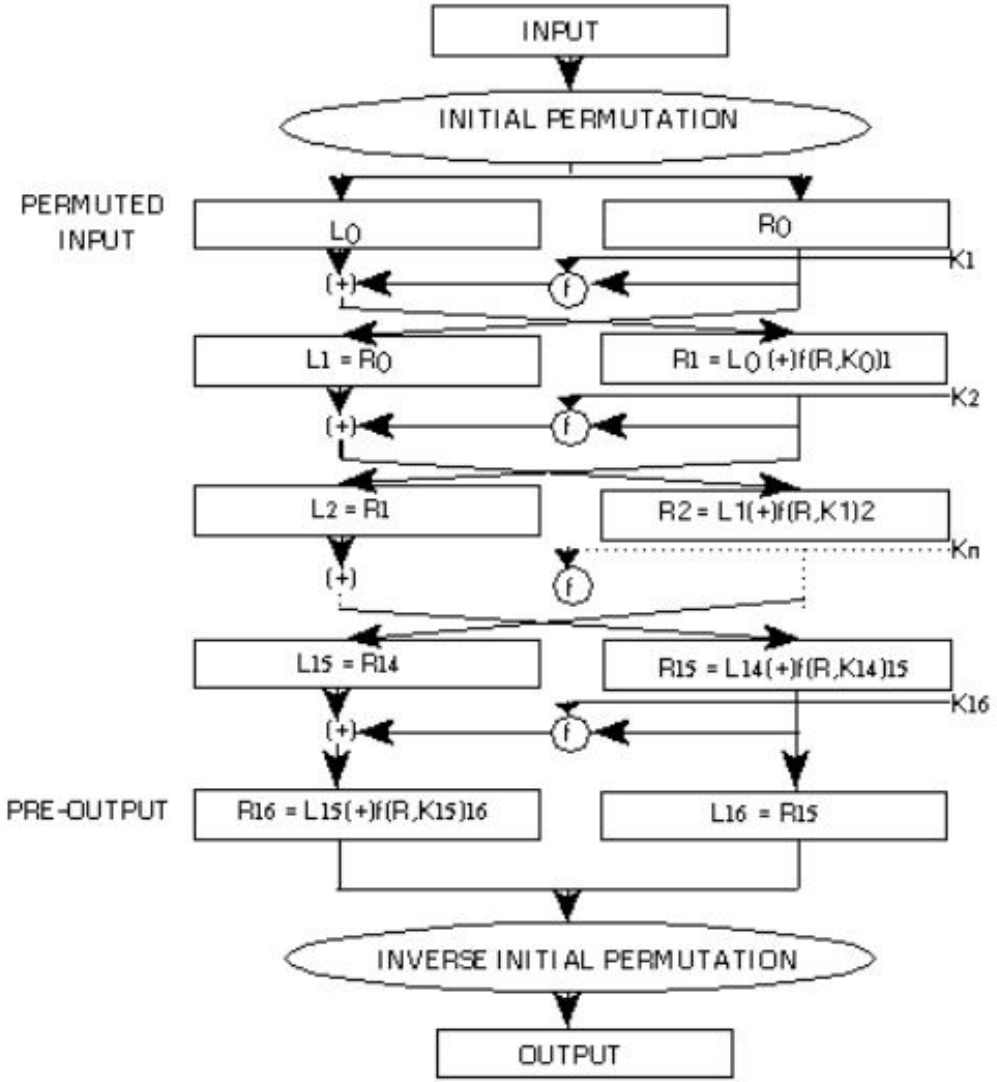
DES - le précurseur

- Première version standard de crypto: marche par bloc de 64 bits avec une clef de 56 bits (plus utilisé).

"After an initial permutation, the block is broken into a right half and a left half, each 32 bits long. Then there are 16 rounds of identical operations, called Function f, in which the data are combined with the key. After the sixteenth round, the right and left halves are joined, and a final permutation (the inverse of the initial permutation) finishes off the algorithm.

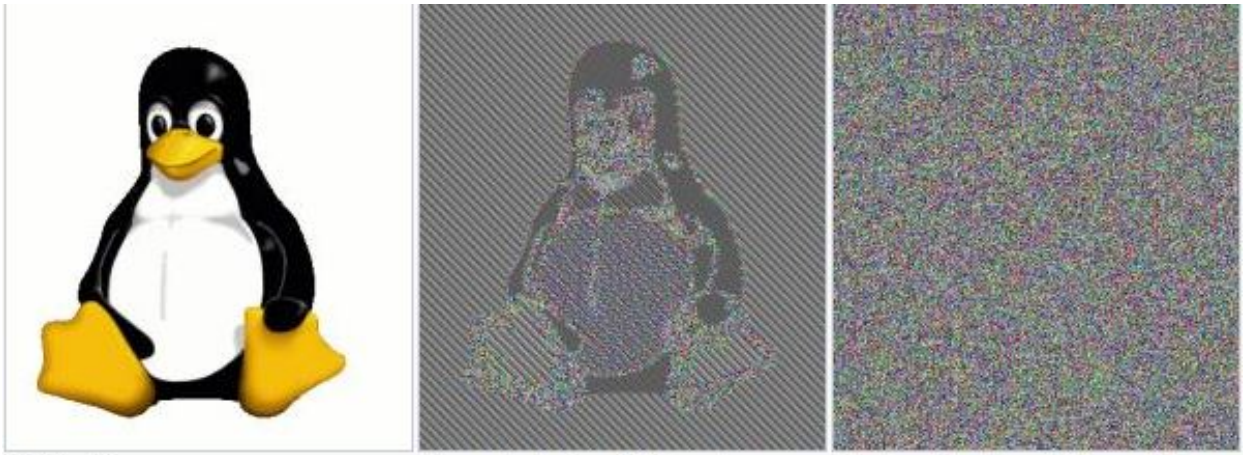
In each round, the key bits are shifted, and then 48 bits are selected from the 56 bits of the key. The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again. These four operations make up Function f. The output of Function f is then combined with the left half via another XOR. The result of these operations becomes the new right half; the old right half becomes the new left half. These operations are repeated 16 times."

DES - visuellement



Mode opérations

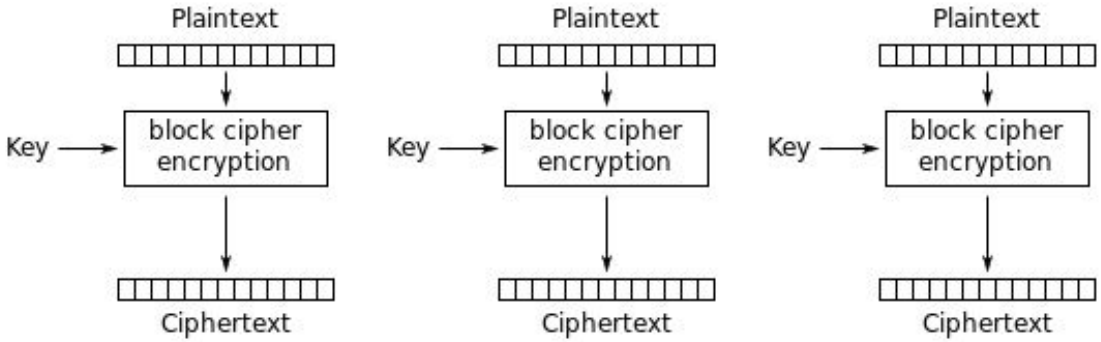
- Il existe de multiples manières d'utiliser un codage par blocs.
- Le but est de masquer des patterns qui pourraient arriver dans les données cryptées (ECB).
- Encryptage d'un bitmap en mode ECB :



Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

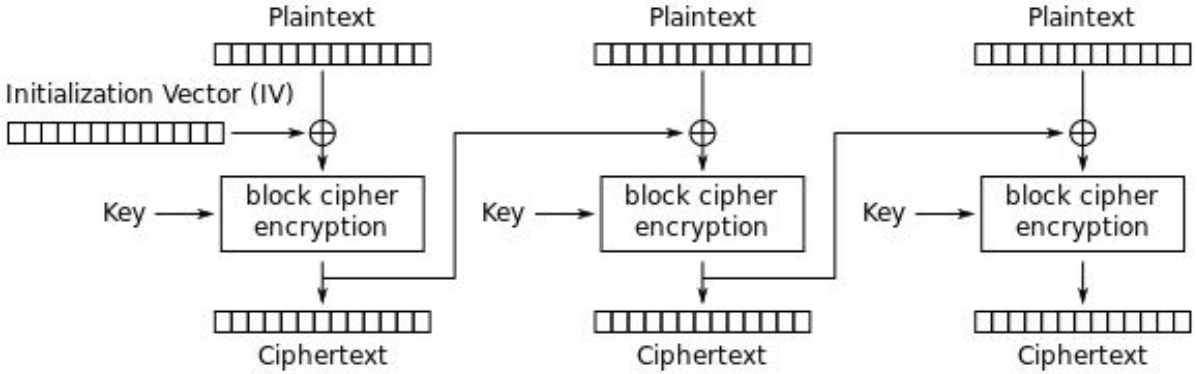
Mode opérations - ECB vs CBC

- ECB



Electronic Codebook (ECB) mode encryption

- CBC



Cipher Block Chaining (CBC) mode encryption

Mode opérations - ECB vs CBC

Mode		Formulas	Ciphertext
Electronic codebook	(ECB)	$Y_i = F(\text{PlainText}_i, \text{Key})$	Y_i
Cipher block chaining	(CBC)	$Y_i = \text{PlainText}_i \text{ XOR } \text{Ciphertext}_{i-1}$	$F(Y, \text{Key})$; $\text{Ciphertext}_0 = IV$
Propagating CBC	(PCBC)	$Y_i = \text{PlainText}_i \text{ XOR } (\text{Ciphertext}_{i-1} \text{ XOR } \text{PlainText}_{i-1})$	$F(Y, \text{Key})$; $\text{Ciphertext}_0 = IV$
Cipher feedback	(CFB)	$Y_i = \text{Ciphertext}_{i-1}$	Plaintext XOR $F(Y, \text{Key})$; $\text{Ciphertext}_0 = IV$
Output feedback	(OFB)	$Y_i = F(Y_{i-1}, \text{Key})$; $Y_0 = F(IV, \text{Key})$	Plaintext XOR Y_i
Counter	(CTR)	$Y_i = F(IV + g(i), \text{Key})$; $IV = \text{token}()$	Plaintext XOR Y_i

AES

- Le successeurs du DES est AES (Rijndel), qui gagne le standard en 2000, et qui existe en multiples versions :
  - Bloc de 128 bits.
  - Clefs de 128, 192 ou 256 bits (10, 12 et 14 tours).
- Avantage de la standardisation:
  - Personne bricole un algo de cryptage dans son coin.
  - Code open source.
- Désavantage de la standardisation:
  - Honey pot.
  - Total break ?

Avantages de l'asymétrie

- A quoi ça sert d'utiliser de la crypto si on peut communiquer les clefs secrètement, autant utiliser le même procédé pour transférer les informations ?
- Problème de gestion des clefs : dans un graphe à n personnes il y a $(n-1)(n-2)/2$ couples possibles.... si le graphe fait 10^6 (petit graphe penser à Facebook ou Twitter par exemple) $\Longrightarrow$ 10^12 clefs ?
- Pourquoi la clef de codage serait elle la même que la clef de décodage ?
  - Cryptage à clef privée/publique.
  - Problème asymétrique : facie de calculer la clef publique à partir de la clef privée mais pas l'inverse.

Protocole Diffie-Hellman

- Première implantation "publique" d'une primitive crypto asymétrique en 1976.

Protocole Diffie-Hellman

- Première implantation "publique" d'une primitive crypto asymétrique en 1976.
- Soit p un nombre premier et g un générateur de $\mathbb{Z}_p$. Ces deux valeurs sont publiques.
- Le secret partagé peut être n'importe quel entier entre 1 et $p-1$.

Protocole Diffie-Hellman

- Première implantation "publique" d'une primitive crypto asymétrique en 1976.
- Soit p un nombre premier et g un générateur de $\mathbb{Z}_p$. Ces deux valeurs sont publiques.
- Le secret partagé peut être n'importe quel entier entre 1 et $p-1$.
- Alice choisit a et envoie $A = g^a \mod p$ à Bob.
- Bob choisit b et envoie $B = g^b \mod p$ à Alice.

Protocole Diffie-Hellman

- Première implantation "publique" d'une primitive crypto asymétrique en 1976.
- Soit p un nombre premier et g un générateur de $\mathbb{Z}_p$. Ces deux valeurs sont publiques.
- Le secret partagé peut être n'importe quel entier entre 1 et $p-1$.
- Alice choisit a et envoie $A = g^a \mod p$ à Bob.
- Bob choisit b et envoie $B = g^b \mod p$ à Alice.
- Alice et Bob calculent respectivement $B^a \mod p$ et $A^b \mod p$.
- Les deux valent la même chose et peuvent êtres utilisés comme clef pour crypter symétriquement.

Le code du sac-à-dos

- Première version simple mais assez simple à casser (des évolutions sont en cours).
- Basé sur le problème du sac-à-dos : étant donnée une liste de valeur $\{v_i \mid i \in 1..n\}$ et un objectif O trouver un sous-ensemble $J \subseteq \{1..n\}$ tel que $\Sigma_{j \in J} = O$.

Le code du sac-à-dos

- Première version simple mais assez simple à casser (des évolutions sont en cours).
- Basé sur le problème du sac-à-dos : étant donnée une liste de valeur $\{v_i \mid i \in 1..n\}$ et un objectif O trouver un sous-ensemble $J \subseteq \{1..n\}$ tel que $\Sigma_{j \in J} = O$.
- Idée du codage : pour coder un vecteur de n bits donne liste des valeurs l et un objectif O, et on prend 1 pour les éléments de la liste qui sont dans la somme et 0 sinon.
  Par exemple si on veut coder 01101 on peut prendre
  $l = \{12, 4, 6, 98, 105\}$ et l'objectif $O = 4 + 6 + 105 = 115$

Le code du sac-à-dos

- Première version simple mais assez simple à casser (des évolutions sont en cours).
- Basé sur le problème du sac-à-dos : étant donnée une liste de valeur $\{v_i \mid i \in 1..n\}$ et un objectif O trouver un sous-ensemble $J \subseteq \{1..n\}$ tel que $\Sigma_{j \in J} = O$.
- Idée du codage : pour coder un vecteur de n bits donne liste des valeurs I et un objectif O, et on prend 1 pour les éléments de la liste qui sont dans la somme et 0 sinon.
  Par exemple si on veut coder 01101 on peut prendre
  $I = \{12, 4, 6, 98, 105\}$ et l'objectif $O = 4 + 6 + 105 = 115$
- Le problème du sac-à-dos est NP-complet ...

Le code du sac-à-dos

- Première version simple mais assez simple à casser (des évolutions sont en cours).
- Basé sur le problème du sac-à-dos : étant donnée une liste de valeur $\{v_i \mid i \in 1..n\}$ et un objectif O trouver un sous-ensemble $J \subseteq \{1..n\}$ tel que $\Sigma_{j \in J} = O$.
- Idée du codage : pour coder un vecteur de n bits donne liste des valeurs l et un objectif O, et on prend 1 pour les éléments de la liste qui sont dans la somme et 0 sinon.
  Par exemple si on veut coder 01101 on peut prendre
  $l = \{12, 4, 6, 98, 105\}$ et l'objectif $O = 4 + 6 + 105 = 115$
- Le problème du sac-à-dos est NP-complet ...
  mais pas si les v_i sont tels que $\sum_{i=1}^{j-1} v_i < v_j$

Le code du sac-à-dos : version asymétrique

- clef privée : préparer une suite supercroissante de valeurs.

$$W = \{w_1, w_2, \ldots, w_n\}$$

Choisir $q > \Sigma_{i=1}^{n} w_i$

Le code du sac-à-dos : version asymétrique

- clef privée : préparer une suite supercroissante de valeurs.

$$W = \{w_1, w_2, \ldots, w_n\}$$

Choisir $q > \sum_{i=1}^{n} w_i$
Choisir un nombre aléatoire r premier avec q, calculer $r' = r^{-1} \mod q$.

- Clef publique : calculer la suite

$$B = \{b_1, b_2, \ldots, b_n\}$$

avec $b_i = rw_i \mod q$

Le code du sac-à-dos : version asymétrique

- clef privée : préparer une suite supercroissante de valeurs.

$$W = \{w_1, w_2, \ldots, w_n\}$$

Choisir $q > \sum_{i=1}^{n} w_i$
Choisir un nombre aléatoire r premier avec q, calculer $r' = r^{-1} \mod q$.

- Clef publique : calculer la suite

$$B = \{b_1, b_2, \ldots, b_n\}$$

avec $b_i = rw_i \mod q$

- cryptage $c = \sum_{i=1}^{n} m_i b_i$ (m_i les bits du message)
- décryptage : $c' := cr' \mod q$
il ne reste qu'à résoudre un problème supercroissant pour retrouver les bits du message.

RSA

- Sûrement l'algo asymétrique le plus utilisé.
- Basé sur l'asymétrie suivante : il est facile de multiplier deux entiers, plus difficile de factoriser (pas d'algo polynomial connu mais pas NP-complet non plus).

RSA

- Sûrement l'algo asymétrique le plus utilisé.
- Basé sur l'asymétrie suivante : il est facile de multiplier deux entiers, plus difficile de factoriser (pas d'algo polynomial connu mais pas NP-complet non plus).
- Algorithme basé sur le petit théorème de Fermat.
  1. Choisir deux nombres premiers p, q. On pose $n = pq$
  2. $\varphi(n) = (p-1)(q-1)$. (valeur indicatrice d'Euler pour n).
  3. Choisir e premier avec $\varphi(n)$.
  4. Calculer d, l'inverse modulo n de e (par l'algo d'Euclide étendu).

RSA

- Sûrement l'algo asymétrique le plus utilisé.
- Basé sur l'asymétrie suivante : il est facile de multiplier deux entiers, plus difficile de factoriser (pas d'algo polynomial connu mais pas NP-complet non plus).
- Algorithme basé sur le petit théorème de Fermat.
  1. Choisir deux nombres premiers p, q. On pose $n = pq$
  2. $\varphi(n) = (p-1)(q-1)$. (valeur indicatrice d'Euler pour n).
  3. Choisir e premier avec $\varphi(n)$.
  4. Calculer d, l'inverse modulo n de e (par l'algo d'Euclide étendu).
- Chiffremen : $M^e \mod n$
- Déchiffrement : $C^d \mod n$

RSA

- Niveau de sécurité (ordinateur classique): pour du 128 bits AES il faut à peu près un 2048 RSA.
  les algos de factorisations progressent !

RSA

- Niveau de sécurité (ordinateur classique): pour du 128 bits AES il faut à peu près un 2048 RSA.
  les algos de factorisations progressent !
- Avantage énorme : la multiplication est commutative ce qui permet la signture !
  On peut calculer $M^e \mod n$ et tout le monde peut le décrypter !
  mais seul celui possédant la clef privée peut le faire.

RSA

- Niveau de sécurité (ordinateur classique): pour du 128 bits AES il faut à peu près un 2048 RSA.
  les algos de factorisations progressent !
- Avantage énorme : la multiplication est commutative ce qui permet la signture !
  On peut calculer $M^e \mod n$ et tout le monde peut le décrypter !
  mais seul celui possédant la clef privée peut le faire.
- Très couteux en temps de calcul.

RSA

- Niveau de sécurité (ordinateur classique): pour du 128 bits AES il faut à peu près un 2048 RSA.
  les algos de factorisations progressent !
- Avantage énorme : la multiplication est commutative ce qui permet la signture !
  On peut calculer $M^e \mod n$ et tout le monde peut le décrypter !
  mais seul celui possédant la clef privée peut le faire.
- Très couteux en temps de calcul.
- Problème des PKI : comment s'assurer que la clef publique corresponde bien à une personne précise (problème de l'identité qui revient tout le temps).

Autres méthodes et crypto post-quantique

- D'autres systèmes existent : El Gamal (problème du log discret, pgp), courbes elliptiques (Ethereum).

Autres méthodes et crypto post-quantique

- D'autres systèmes existent : El Gamal (problème du log discret, pgp), courbes elliptiques (Ethereum).
- Les ordinateurs quantiques peuvent factoriser en temps polynomial : RSA est craqué

Autres méthodes et crypto post-quantique

- D'autres systèmes existent : El Gamal (problème du log discret, pgp), courbes elliptiques (Ethereum).
- Les ordinateurs quantiques peuvent factoriser en temps polynomial : RSA est craqué
- Des candidats pour la crypto post quantique (taille des clefs pour un équivalent sécurité AES 128):
  - NTRU : réseaux euclidiens et problème du plus court vecteur. Clef publique 6000 bits.
  - polynômes multivariés : 991 000 bits de clef publique.
  - Codes correcteurs : 8 Mégas pour la clef publique.

Concrètement comment crypter ?

- Mieux vaut ne pas réimplanter la poudre (efficacité, erreurs etc.). Il existe des implantations open source (donc dont tout le monde peut valider l'implantation) d'à peu près tous les algos présentés (et si l'algo n'a pas une implantation connue c'est louche).
- La majorité des "bons" systèmes de cryptage sont implantés dans `openssl enc`, y compris RSA.
- Pour récupérer de l'aléatoire : `/dev/random`
- Les "bonnes" fonctions de hachage sont aussi implantées dans `openssl dgst`.